FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico Final
Jazz Jackrabbit 2

[75.42] Taller de Programación I
Primer Cuatrimestre de 2024

Estudiante	**Padrón**	**Email**
Buono, Fernando	103523	fbuono@fi.uba.ar
Duca, Francisco	106308	fduca@fi.uba.ar
Oshiro, Lucas	107024	loshiro@fi.uba.ar
Shiao, Tomás Jorge	106099	tshiao@fi.uba.ar

Índice

1. Organización del Proyecto

Para el desarrollo del proyecto, se decidió dividir el trabajo entre cliente y servidor. A su vez, en el lado del cliente, se subdividieron las tareas entre QT y SDL, Lobby y Game play.

1.1. Servidor

Fernando y Francisco se encargaron de la implementación de la lógica del juego, así también como la comunicación con el cliente. Para ello, se diseñó un protocolo de comunicación entre cliente y servidor, que permite la sincronización de los estados del juego. Para esto, se utilizan sockets, threads y monitores para garantizar la correcta sincronización de los datos.

A través de los sockets, se envían y reciben DTOs (Data Transfer Objects). Estos pueden ser para la creación de una partida, para indicar el comienzo de un juego, para informar el estado actual de la partida, entre otros.

La comunicación con el cliente se puede subdividir en dos grandes áreas: el Lobby y el Juego. En el Lobby, se realiza un ping-pong en su mayor parte, donde el cliente solicita la creación de una partida, las partidas disponibles para unirse, los mapas disponibles o unirse a una partida. La única instancia en que no es un ping pong es cuando el cliente solicita jugar en una partida multijugador que no está completa: en este caso, se queda esperando por mensajes y al unirse un nuevo jugador, el servidor realiza un broadcast del mismo. En el Juego, en tanto, constantemente se envían mensajes de actualización de estado, para mantener sincronizados a los jugadores. De este modo, todos pueden ver qué es lo que está ocurriendo en la partida en tiempo real.

1.2. Cliente

El cliente se inicia con el Lobby, donde el jugador selecciona qué es lo que desea hacer: crear una nueva partida o unirse a una de ellas. Luego, cuando inicia el juego, se pasa al gameplay. Posee dos controladores, una para cada sección, que manejan las comunicaciones con el servidor. Estos, al tener que enviar un mensaje, lo serializan y luego se lo pasan al thread emisor, que lo envían via el socket de comunicación. El thread receptor, en tanto, se encuentra recibiendo los mensajes del servidor, y al recibir uno, lo deserializa y se lo pasa al controlador.

1.2.1. SDL

Lucas fue quien estuvo trabajando con SDL. Para ello, se creó un motor de juego que permite la creación de escenarios, personajes, enemigos, armas, municiones, gemas, monedas, entre otros elementos. A su vez, se crearon las animaciones de los personajes, los sonidos y la música ambiente a partir de los sprites y audios provistos por la cátedra.

La comunicación con el servidor, como fue explicado anteriormente, es constante y se realiza a través del controlador del juego. Recibe estados del juego, conteniendo las posiciones de todos los elementos e información acerca de cada uno de ellos, tales como la cantidad de vidas, puntajes, el estado de cada jugador o enemigo, entre otros. Es a partir de estos datos que se renderiza cada frame del juego, logrando así una sincronización entre el cliente y el servidor y la visualización de la partida en tiempo real.

1.2.2. QT

Lobby Por último, Tomás fue quien se encargó de realizar el Lobby. Para ello, se utiliza QT5, una herramienta que permite la creación de interfaces. En el Lobby, se pueden crear partidas, unirse a partidas, ver las partidas disponibles, ver los mapas disponibles y seleccionar el personaje con quien se quiere jugar.

A medida que se pasan las pantallas, el Lobby va completando una estructura LobbyMessage, que contiene los datos que se le envían al servidor a través del controlador. En él, se encuentran

los datos de la partida, el mapa, el personaje, entre otros datos, dependiendo de lo que seleccione el cliente. El controlador envía y recibe los mensajes desde el servidor como fue explicado anteriormente. En el momento que se puede empezar la partida, cuando se alcanzan la cantidad de jugadores necesarios, el servidor envía un mensaje de inicio de juego, y el cliente cambia de pantalla, pasando al gameplay.

Editor de Niveles Para el editor de niveles también se usó QT5. En este programa se pueden seleccionar los distintos elementos del juego y diseñar un nuevo mapa. Para ello, se tiene una columna con todos los elementos y un canvas sobre al cual se le pueden arrastrar los elementos encima. Una vez que el mapa se guarda, el editor procesa los elementos sobre el canvas y guarda sus posiciones en un archivo YAML en el servidor.

2. Features

Feature	**Estado**	**Comentario**
Acciones Comunes	En Progreso	
Estados	En Progreso	
Acciones Especiales	Por Implementar	
Armas y Municiones	En Progreso	
Gemas y Monedas	Implementado	
Enemigos	En Progreso	
Escenarios	Implementado	
Multijugador	En Progreso	
Cámara	En Progreso	
Animaciones	Implementado	
Sonidos y Música Ambiente	Implementado	
Interfaz del Jugador	En Progreso	Queda pendiente el top 3 de puntajes.
Configuración YAML	Implementado	
Interfaz del Lobby	Implementado	
Editor de Niveles	En Progreso	No permite selección de fondo ni spawn points.
Vagrant	Implementado	

3. Known Issues

-